Una técnica para una especificación inicial en RSL

Virginia Mauco[*] Daniel Riesco[†]

[*] **Universidad Nacional del Centro de la Provincia de Buenos Aires**
San Martín 895
7000 Tandil – Argentina

[†] **Universidad Nacional de San Luis**
Departamento de Informática
Ejército de los Andes 950
5700 San Luis - Argentina

1. Introducción

Los métodos formales ayudan a incrementar la confiabilidad del software. Estos métodos permiten "razonar" acerca de propiedades del software o de sistemas que involucran software. Además, ofrecen la seguridad de que los requisitos son verificados en cada paso del desarrollo, encontrando inconsistencias e incompletitud. Las especificaciones formales se pueden usar durante todo el ciclo de vida del software y se pueden manipular con herramientas automáticas con una amplia variedad de propósitos tales como *model checking*, verificación deductiva, animación, generación de casos de prueba, reuso formal de componentes, y refinamiento de especificación a implementación. El Método RAISE (*Rigourous Approach to Industrial Software Engineering*) [3], por ejemplo, provee facilidades para el uso industrial de métodos formales en el desarrollo de sistemas de software. Este método provee de un gran número de técnicas y estrategias para hacer desarrollos formales y pruebas, además de un lenguaje formal de especificación, *RAISE Specification Language* (RSL) [2], y un conjunto de herramientas. Sin embargo, estos métodos son, en general, sólo accesibles a especialistas debido a que sus formalismos matemáticos son más difíciles de entender y comunicar.

Esto es particularmente inconveniente durante las primeras etapas de desarrollo, donde la interacción con los *stakeholders*, no familiarizados con estas notaciones, es crucial para el desarrollo exitoso del sistema. Los requisitos de un sistema se deben describir de manera tal que sea posible llegar a un acuerdo entre los *stakeholders* y el equipo de desarrollo sobre qué hará y qué no hará el sistema. Un desafío importante es que los *stakeholders* deben ser capaces de leer y entender los resultados de la captura de requisitos, y para esto es necesario usar el lenguaje de los *stakeholders* para describir esos resultados. Más aún, los dominios son naturalmente informales porque residen en el mundo real. Un buen enfoque formal, necesaria e inevitablemente, usa técnicas formales e informales [1].

Por esto, sería muy útil el uso de técnicas orientadas a mejorar la comunicación entre los *stakeholders* y los ingenieros de software. Estas técnicas son muy útiles para obtener una primera especificación de un sistema, que podrá ser fácilmente validada con los *stakeholders* y que podría ser la base para un desarrollo formal. De esta manera, se aprovecharían las ventajas de ambas técnicas en las diferentes etapas de desarrollo para mejorar el producto final. Sin embargo, será necesario encontrar alguna manera de establecer *mappings* entre el mundo conceptualmente más rico de la ingeniería de requisitos y el mundo de los métodos formales.

Entre los mecanismos propuestos para formalizar la obtención de requisitos, se puede mencionar la *Requirements Baseline* [4,5]. El Modelo Léxico, implementado por el Léxico Extendido del Lenguaje (LEL), y el Modelo de Escenarios son dos de los modelos de esta *Requirements Baseline*. Ambos modelos proveen una descripción detallada de un dominio de aplicación y, como se describen utilizando lenguaje natural, son accesibles a todos los involucrados en el proceso de desarrollo, facilitando así la comunicación entre *stakeholders* e ingenieros de software. Pero un punto importante es cómo aprovechar y utilizar toda la información contenida en estos modelos a lo largo del proceso de desarrollo de software.

Para encontrar alguna solución a los problemas mencionados anteriormente, en este proyecto de investigación proponemos analizar y desarrollar la integración de técnicas de Ingeniería de Requisitos con especificaciones formales en RSL. Se pretende desarrollar una técnica que permita obtener una especificación inicial en RSL partiendo de modelos de requisitos, tales como LEL y escenarios que están más próximos al lenguaje de los *stakeholders*. De esta manera, contribuimos a disminuir el *gap* existente entre los *stakeholders* y los ingenieros especialistas en métodos formales. Una vez derivada esta especificación inicial, el proceso de desarrollo de software continúa con las etapas propuestas en el Método RAISE.

Para realizar este estudio, elegimos el dominio de los Sistemas de Producción Bovina de Leche, una de las componentes de la infraestructura Sistemas Agropecuarios.

2. Requirements Baseline

Entre los mecanismos propuestos para formalizar la obtención de requisitos, se puede mencionar la *Requirements Baseline* [4,5] que permite conocer y registrar los orígenes de los requisitos y evoluciona durante el proceso de desarrollo de software. Es una estructura que incorpora descripciones sobre el dominio del problema y el software a construir sobre ese dominio. Está compuesta por cinco vistas, dos de las cuales son el Modelo Léxico y el Modelo de Escenarios.

El Modelo Léxico está implementado por el Léxico Extendido del Lenguaje (LEL). El LEL es una estructura que permite la representación de los términos significativos del dominio estudiado, delimitando el lenguaje externo y enriqueciendo el interno. Se construye usando lenguaje natural. Cada término define un objeto (entidad pasiva), un sujeto (entidad activa), una frase verbal o un estado/situación. El objetivo del LEL es comprender el lenguaje del problema sin preocuparse por entender el problema.

El Modelo de Escenarios tiene por objetivo modelar aspectos de comportamiento. Un escenario es una descripción parcial de un comportamiento dentro del dominio que ocurre en un momento dado y en un contexto geográfico específico. Aunque cada escenario describe una situación particular, ninguno es completamente independiente de los demás, sino que por el contrario mantiene una relación semántica con otros escenarios. Para la representación de los escenarios, también se utiliza lenguaje natural.

La definición de requisitos implica la aplicación de un lenguaje. Hasta no hace mucho tiempo, el lenguaje del diseñador dominaba el proceso de definición de requisitos, afectando en gran medida la validez de los mismos. Sin embargo, es necesario realizar este proceso de definición en términos del lenguaje manejado por los *stakeholders*.

El LEL y los escenarios permiten a los *stakeholders* acceder al proceso de definición de requisitos ya que, al estar construidos usando lenguaje natural, pueden ser comprendidos por todos los involucrados en el desarrollo.

3. El Método RAISE

RAISE es un método formal que provee facilidades para el uso industrial de Métodos Formales en el desarrollo de sistemas de software [3]. Consiste de un método de desarrollo, un lenguaje formal de especificación y un conjunto de herramientas.

El objetivo general del método es desarrollar una secuencia de especificaciones de un sistema, donde una especificación es una colección de módulos RSL y su traducción en un lenguaje de programación. La primera especificación (especificación inicial) es generalmente abstracta aplicativa, es decir los módulos contienen tipos abstractos de datos y signaturas y axiomas, en lugar de definiciones explícitas, para algunas o todas las funciones.

El método de desarrollo está basado en el paradigma de refinamientos por pasos sucesivos. De acuerdo a este paradigma, el software se construye mediante una serie de etapas, cada una de las cuales es un refinamiento de la anterior. En RAISE, cada etapa se produce con la filosofía *invent and verify*, es decir una especificación se desarrolla en un diseño “manualmente” (el aspecto de invención) y luego se verifica la relación entre los dos. La mayor parte del método de desarrollo consiste de técnicas para los cuatro procedimientos principales: especificación, desarrollo, justificación y traducción.

El lenguaje de especificación RSL es un lenguaje de especificación de amplio espectro, en el sentido que se puede usar para expresar especificaciones abstractas de alto nivel así como también diseños de bajo nivel (por ejemplo, usando construcciones imperativas explícitas). Una ventaja de esto, es que los usuarios sólo deben conocer una notación, además del lenguaje de programación usado para la implementación, ya que el desarrollo completo se realiza en RSL. Las facilidades de estructuración de RSL soportan descomposición y reuso.

4. Objetivo del proyecto

El objetivo general de este proyecto es analizar y desarrollar la integración de técnicas de Ingeniería de Requisitos con especificaciones formales en RSL, haciendo énfasis en la Ingeniería de Dominio. Las etapas para alcanzar este objetivo son las siguientes:

1) Desarrollo del LEL y Modelo de Escenarios para un dominio específico.

2) Definición de la especificación inicial del dominio elegido en RSL.

3) Análisis del LEL, Modelo de Escenarios y especificación inicial en RSL para definir un conjunto de heurísticas que permitan obtener la especificación inicial a partir de los dos primeros.

4) Generalización de la estrategia definida en la etapa anterior para definir una técnica que se pueda aplicar a cualquier dominio.

5. Estado actual

La ventaja de los métodos formales como RAISE es que ayudan a evitar la ambigüedad e interpretaciones incorrectas de los requerimientos. Además, aseguran un proceso de desarrollo correcto en base a demostraciones o pasos correctos por construcción. Sin embargo, como ya hemos expresado, estos métodos son difíciles de entender para los *stakeholders*.

La propuesta de investigación que presentamos aquí apunta a solucionar este problema, ya que permitiría la derivación de una especificación inicial en RSL partiendo de dos de los modelos de la *Requirements Baseline*.

Hasta el momento, hemos desarrollado un proceso de definición para una especificación inicial en RSL que consta de tres pasos: derivación de tipos, derivación de módulos y derivación de funciones. Definimos también un conjunto preliminar de heurísticas que muestran cómo derivar tipos y funciones, y cómo estructurarlos en módulos, partiendo de descripciones de un dominio escritas en lenguaje natural como LEL y escenarios. Estas heurísticas son guías acerca de cómo comenzar con la definición de una especificación formal, partiendo de la gran cantidad de información contenida en LEL y escenarios. Una descripción detallada se puede encontrar en [6].

El proceso propuesto se aplicó a un caso de estudio completo, el dominio de los Sistemas de Producción Bovina de Leche. Para este dominio construimos, previamente, el Modelo Léxico y el Modelo de Escenarios. Para construir el LEL, realizamos entrevistas estructuradas y no estructuradas con expertos del dominio. Como resultado, se obtuvo una lista con 70 términos, cada uno de los cuales fue validado por los expertos del dominio. Los escenarios se construyeron siguiendo una estrategia combinada. En primer lugar, se produjo una lista de escenarios candidatos a partir del LEL, siguiendo las heurísticas propuestas en [5], la que luego se completó y mejoró para obtener la lista final que contiene 32 escenarios y que fue completamente validada por los expertos del dominio.

Referencias

[1] Bjorner, Dines. “Software Engineering: A New Approach”. Lecture Notes. Technical University of Denmark. 2000. www.it.dtu.dk/~db.

[2] George Ch., Haff P., Havelund K., Haxthaunsen A., Milne R., Nielsen C., Prehn, S. and Ritter, K. “The RAISE Specification Language”. Prentice Hall. 1992.

[3] George Ch., Haxthausen A., Hugues S., Milne R., Prehn S. and Pedersen J.S. “The RAISE Development Method”. Prentice Hall.

[4] Leite, J.; Oliveira, A. “A Client Oriented Requirements Baseline”. Proceedings of the Second IEEE International Symposium On Requirements Engineering. 1995. pp 108-115.

[5] Leite, J.; Hadad, G.; Doorn, J.; Kaplan, G. “A Scenario Construction Process”. Requirements Engineering Journal. Vol 5, Nº 1, pp 38-61. Springer Verlag. 2000.

[6] Mauco, V.; George, Ch. Using requirements engineering to derive a formal specification. Research Report 223, United Nations University/International Institute for Software Technology, Macau. Diciembre 2000.